Script to find radical expressions for p-th roots of unity

Eric Binnendyk

August 17, 2019

1 Introduction and History

In this document, I describe a Python program that I wrote to calculate expressions for the pth roots of unity in radicals, where p is prime.

The result presented here derives heavily from earlier research.

The study of radical expressions for roots of unity began with Gauss's discovery of an expression for 17th roots of unity using only square roots [2]. Gauss went on to prove that all roots of unity could be expressed in radicals and developed a method to find such expressions [3]. A similar method was also developed by Vandermonde [4]. The algorithm I describe here is based on Gauss's method.

Later, Galois developed a theory of solvable algebraic equations. One result of this theory is that any polynomial whose Galois group is Abelian is solvable. Applying this result to cyclotomic polynomials leads to a second proof of Gauss's result of the existence of radical expressions for roots of unity.[5]

Others have developed algorithms based on Gauss's method to find radical expressions for roots of unity, including Andreas Weber [6] and Lau Jing Feng [7]. However, many of these algorithms evaluated multisums such as $T_1 = \sum_{k=0}^{q-1} \sum_{j=0}^{p-1} B_{kl} x^l y^k$ by solving for the values $\sum_{j=0}^{p-1} B_{kl} x^l$ and substituting in known expressions for the qth roots of unity y^k. This method can be refined by simplifying the entire expression as if it were a single qth root of unity, into pieces expressed as sums of pth roots of unity, which can then further be simplified into nested radical expressions. The refined method avoids performing direct arithmetic on large and cumbersome radical expressions for yth roots of unity, and in addition the output expressions tend to be shorter. However, Weber's algorithm has the benefit that it only requires one calculation to compute the values $T_1, T_2, \ldots, T_{q-1}$ (in my notation), an optimization which I have not implemented into my algorithm yet.

2 Preliminaries

An intuitive way to understand algebraic numbers is by expressing them in radicals. Such expressions consist of sums of rational numbers and nth roots of simpler radical expressions, where n can be any positive integer. The Abel-Ruffini theorem says that polynomials of degree up to 4 with rational coefficients have radical expressions for their roots. Some higher degree polynomials also have radical expressions for their roots.

An nth *root of unity* is a complex number of the form $e^{2k\pi i/n} = \cos 2k\pi/n + i \sin 2k\pi/n$ for some integer k where $0 \leq k < n$, with a magnitude of 1 and an argument k/n of a full 2π radians. A *primitive* nth root of unity requires additionally that k and n be relatively prime. As a consequence of this definition, a primitive nth root of unity is not an mth root of unity for any $m < n$.

If x is an nth root of unity, then x satisfies the polynomial $x^n - 1$. This polynomial is factorizable as $(x-1)(\sum_{k=0}^{n-1} x^k)$. Thus, if $x \neq 1$, x also satisfies $\sum_{k=0}^{n-1} x^k$. A polynomial satisfied uniquely by primitive nth roots of unity, for some n, is called a *cyclotomic polynomial*. If p is prime, then all pth roots of unity besides 1 are primitive. Thus, the expression $\sum_{k=0}^{p-1} x^k$ is satisfied by all, and only by, primitive pth roots of unity. It follows that for prime p, the pth cyclotomic polynomial is given by $\sum_{k=0}^{p-1} x^k$.

We know that radical expressions exist for all roots of unity. For example, let p be prime. To get the pth roots of unity, we can just write $\sqrt[p]{1}$ for all p choices of the complex pth root. However, this expression is not desirable for two reasons:

1. It does not only describe the $p-1$ primitive pth roots but also describes 1, obtained in the previous expression when $i = 0$.
2. It does not separate into radical expressions for the real and imaginary parts.

An example of an expression that does not have these two problems is

$$e^{2\pi i/5} = \frac{-1 + \sqrt{5} + \sqrt{-10 - 2\sqrt{5}}}{4}$$

This is an expression for a primitive 5th root of unity as the root of the cyclotomic polynomial $x^4 + x^3 + x^2 + x^1 + 1$, which is satisfied by all four primitive 5th roots of unity. Changing the signs on this expressions yields the other three primitive 5th roots, but not 1.

$$e^{4\pi i/5} = \frac{-1 - \sqrt{5} + \sqrt{-10 + 2\sqrt{5}}}{4}$$

$$e^{6\pi i/5} = \frac{-1-\sqrt{5}-\sqrt{-10+2\sqrt{5}}}{4}$$

$$e^{8\pi i/5} = \frac{-1+\sqrt{5}-\sqrt{-10-2\sqrt{5}}}{4}$$

By a theorem of Galois, it follows from the fact that the Galois group of pth roots of unity is always Abelian that radical expressions describing only primitive roots of unity must exist for the pth roots of unity, for all primes p. In fact, this existence was known before Galois and was published by Gauss in *Disquisitiones Arithmeticae* [1]. My program generates such expressions, and here I describe the underlying algorithm.

3 Symmetric sums

The first step in the algorithm is to arrange the primitive roots of unity into an order called a *power cycle*. Such a permutation works by starting with x, where x can be any primitive root of unity but is typically $e^{2\pi i/p}$, with an argument of $1/p$ of a circle — the first root encountered when going counterclockwise around the complex plane from 1. Because $x^p = 1$ (and $x^k \neq 1$ for all $k < p$), the full set of $p-1$ primitive roots of unity can be assigned the following *canonical expressions*: $x^1, x^2, x^3, \ldots, x^{n-1}$. To permute these roots into a power cycle, we write $x^1, x^n, x^{n^2}, x^{n^3}, \ldots, x^{n^{p-2}}$, etc. for a special integer n. These roots have the canonical expressions $x^1, x^{n \mod p}, x^{n^2 \mod p}, \ldots, x^{n^{p-1} \mod p}$. We need to choose n such that the cycle contains every primitive root once. Equivalently, the exponents $1, n \mod p, n^2 \mod p, \ldots, n^{p-1} \mod p$ must all be distinct. A value of n satisfying this requirement is called a *primitive root* modulo p (not to be confused with a primitive root of unity, which is an entirely different concept).

For example, for $p = 13$, we see that $n = 2$ is a primitive root mod 13 leading to the following power cycle:

$$x^1, x^2, x^4, x^8, x^3, x^6, x^{12}, x^{11}, x^9, x^5, x^{10}, x^7$$

(In contrast, $n = 3$ is not a primitive root mod 13; the cycle x^1, x^3, x^9 has period 3 before repeating to $x^{9\times 3 \mod 13} = x^1$.)

In any sum of pth roots of unity with integer coefficients, define a *conjugate permutation* to be the operation that replaces all terms of the form ax^k in the sum by $ax^{nk \mod p}$, where n is the same primitive root mod p chosen above. For example, when $p = 13$, the conjugate permutation of $x^1+x^8+x^{10}$ is $x^2+x^3+x^7$. Integers by themselves are self-conjugate, so the conjugate of $2+3x^{11}$ is $2+3x^9$.

Typically, the conjugate permutation of an expression needs to be taken $p-1$

times to return to the same expression, as that is the period of the power cycle. However, in some symmetric expressions, the conjugate can be taken fewer than $p-1$ times to obtain the same expression. For example, when $n = 13$, the expression $x^2 + x^6 + x^5$ has period 4:

$$x^2+x^6+x^5 \to x^4+x^{12}+x^{10} \to x^8+x^{11}+x^7 \to x^3+x^9+x^1 \to x^6+x^5+x^2 = x^2+x^6+x^5$$

We see that conjugation replaced each exponent with the one four terms ahead in the power cycle. Because the expression was invariant under this transformation, the same term with the shifted exponent must have been in the original expression, as well as the one with the exponent shifted by 8. Shifting by 12 wraps the exponent around to its original value, so these are the only three terms needed for the expression to be invariant under four conjugate permutations.

The minimum number of conjugate permutations that are invariant on a particular sum of pth roots of unity is a factor f of $p-1$, and for each exponent k present in the original expression, there is an equivalent term with the exponent $kn^f \mod p$. Repeating this process reveals that non-integer terms must come in sets of $(p-1)/f$, and the sum can be said to have a *symmetry* of $(p-1)/f$.

Note that in the extreme case, an expression can have a symmetry of $p-1$, in which case it has the form $a + \sum_{k=1}^{p-1} bx^k$. But the primitive pth roots of unity satisfy the polynomial $\sum_{k=0}^{p-1} x^k$, so the sum can be rewritten as $a - b$. Therefore, any expression with symmetry $p-1$ evaluates to an integer.

The above definition defines the symmetry of an expression of the form $a_0 + a_1x^1 + a_2x^2 + \cdots + a_{p-1}x^{p-1}$, where $a_0, a_1, \ldots, a_{p-1}$ are integers. We can define a *multisum* of roots of unity to be a sum of products of qth roots of unity for different primes q. Such an expression has the form

$$S = \sum_{k_1,k_2,k_3,\ldots} (a_{k_1,k_2,k_3,\ldots}(x_1)^{k_1}(x_2)^{k_2}(x_3)^{k_3}\ldots)$$

where $x_1 = e^{2\pi/q_1}, x_2 = e^{2\pi/q_2}, x_3 = e^{2\pi/q_3}, \ldots$ are roots of unity whose degrees are distinct primes $q_1, q_2, q_3, \ldots$ and the $a_{k_1,k_2,k_3,\ldots}$ values can be any integers. We can define the symmetry of the above multisum S with respect to one of the primes, say q_1. First we rewrite S as follows:

$$\sum_{k_1=0}^{q_1-1} (A_{k_1}(x_1)^{k_1})$$

where

$$A_{k_1} = \sum_{k_2,k_3,\ldots} (a_{k_1,k_2,k_3\ldots}(x_2)^{k_2}(x_3)^{k_3}\ldots)$$

Then the symmetry with respect to q_1 can be determined by checking for the existence of cycles of the form $A_k(x_1)^k$ whose coefficients A_k are equal.

It is clear that the same multisum can have different values of symmetry with respect to different primes $q_1, q_2, \ldots$, depending on the values of the coefficients that appear when we pull different roots of unity out of the equation (...).

Just like how a sum of pth roots with a symmetry of $p-1$ can be simplified into an integer, a multisum with a symmetry of $q_1 - 1$ with respect to q_1st roots can be simplified by eliminating q_1st roots from the expression, as follows:

$$\sum_{k_1=0}^{q_1-1} (A_{k_1}(x_1)^{k_1}) = A_0(x_1)^0 + \sum_{k_1=1}^{q_1-1} A_1(x_1)^{k_1} \text{ (by symmetry)}$$

$$= A_0 + A_1 \sum_{k_1=1}^{q_1-1} (x_1)^{k_1}$$

$$= A_0 + A_1 \cdot (-1)$$

$$= A_0 - A_1$$

This results in a multisum of q_2nd, q_3rd, ... roots only.

Because sums and multisums with a symmetry of $p-1$ can be simplified by eliminating pth roots of unity, we may imagine that expressions with higher symmetry are "simpler" than those with lower symmetry. This intuition will be validated when we see that higher symmetries indeed lend themselves to shorter expressions in radicals than do lower symmetries.

4 Crux of the algorithm

The crucial step in my algorithm is a way to simplify a sum in terms of sums with higher symmetry.

Consider a sum A_0 of pth roots of unity with symmetry $(p-1)/f$, $f > 1$. Let q be any prime factor of f. The objective of this procedure is to develop sums and multisums with a symmetry of $q(p-1)/f$ with respect to the pth roots, and express the sum A_0 in terms of them.

Apply the conjugate permutation f/q times to obtain a new sum A_1. Repeat this q times to obtain sums $A_2, A_3, \ldots, A_{q-1}$ before reaching A_0 again. The base conjugate permutation gets applied f/q times for q steps, for a total of f applications. Because A_0 has a symmetry of p/f, the f successive applications of the conjugate permutation result in a return to A_0 at the end.

4.1 General simplification procedure

Clearly the sum $S_0 := \sum_{k=0}^{q-1} A_k$ has a symmetry of $(p-1)/(f/q) = q(p-1)/f$, as applying the conjugate permutation f/q times will cyclically permute the terms. Now consider the $q-1$ multisums $S_j := \sum_{k=0}^{q-1} y^{jk} A_k$, where $1 \le j \le q-1$ and y is a primitive qth root of unity. The sequence $S_0, S_1, S_2, \ldots S_{q-1}$ forms the *discrete Fourier transform* of the terms A_k.

The sum S_0, with a symmetry q times that of A_0, is already simpler than A_0. Now we must express $S_1, S_2, \ldots, S_{q-1}$ in terms of simpler sums.

4.1.1 The qth power

Consider $(S_1)^q$, which is equal to $(\sum_{k=0}^{q-1} y^k A_k)^q$. This can be expanded, grouping alike powers of y, to yield the sum

$$\sum_{k=0}^{q-1} B_k y^k$$

where:

$$\begin{aligned} B_0 = A_0 \cdot (q-3 \text{ factors omitted}) \cdot A_0 \cdot A_0 \\ + A_0 \cdot (q-3 \text{ factors omitted}) \cdot A_1 \cdot A_{q-1} \\ + A_0 \cdot (q-3 \text{ factors omitted}) \cdot A_2 \cdot A_{q-2} \\ + \ldots \\ + A_{k_0} \cdot \cdots \cdot A_{k_{q-1}} \text{ where } k_0 + \cdots + k_{q-1} \mod q = 0 \end{aligned}$$

and in general:

$$B_j = \sum_{k_0,\ldots,k_{q-1}} \prod_{n=0}^{q-1} A_{k_n}$$

where $k_0, \ldots, k_{q-1}$ ranges over all choices such that $k_0 + \cdots + k_{q-1} \mod q = j$.

The values $A_0 \ldots A_{q-1}$ are sums of pth roots of unity with symmetry $(p-1)/f$. Because symmetry is preserved under multiplication of sums (the proof is omitted), the expressions $\prod_{n=0}^{q-1} A_{k_n}$ also have symmetry of $(p-1)/f$. In fact, if some B_j contains the term

$$\prod_{n=0}^{q-1} A_{k_n}$$

in its sum, for some k_n, it will also contain the terms

$$\prod_{n=0}^{q-1} A_{k_n+1} \mod q$$

$$\prod_{n=0}^{q-1} A_{k_n+2} \mod q$$

all the way up to

$$\prod_{n=0}^{q-1} A_{k_n-1} \mod q$$

as the subscripts on these terms sum to the same value, modulo q.

Applying a base conjugate permutation f/q times to each term above yields the next term, with the last term wrapping back around to the first. (Again, the proof is omitted.) Thus, the sum of the q terms is invariant under the operation and has a symmetry of $q(p-1)/f$. Because each B_j consists entirely of such sums, each B_j has a symmetry of $q(p-1)/f$ as well.

Thus, S_1^q, despite being a multisum rather than a sum, is simpler than A_0 because it has a symmetry of $q(p-1)/f$ rather than $(p-1)/f$ with respect to $p-1$.

By equivalent arguments, it can be shown that $S_2^q, \ldots, S_{q-1}^q$ also have a symmetry of $q(p-1)/f$. From now on, we will call these expressions $T_1, T_2, \ldots, T_{q-1}$.

4.1.2 Back to A_0

We now have q expressions that are simpler than A_0:

$$S_0 = A_0 + A_1 + \cdots + A_{q-1}$$

$$T_1 = (A_0 + y^1 A_1 + \cdots + y^{q-1} A_{q-1})^q$$
$$T_2 = (A_0 + y^2 A_1 + \cdots + y^{q-2} A_{q-1})^q$$
$$T_3 = (A_0 + y^3 A_1 + \cdots + y^{q-3} A_{q-1})^q$$

all the way up to

$$T_{q-1} = (A_0 + y^{q-1} A_1 + \cdots + y^1 A_{q-1})^q$$

Using the *inverse discrete Fourier transform* we can obtain values for $A_0, A_1, \ldots, A_{q-1}$. In particular,

$$A_0 = \frac{1}{q}(S_0 + \sqrt[q]{T_1} + \cdots + \sqrt[q]{T_{q-1}})$$

for some choices of complex qth root. The choice of root needed can be determined by finding a complex floating-point expression for S_k and comparing it to all q choices of qth root of T_k.

The expressions $T_1 \ldots T_{q-1}$ have symmetry of 1 with respect to qth roots of unity. They can be simplified by increasing the symmetry of qth roots as described below in "Extension to multisums", eventually allowing the qth roots to

be removed so that simplification on the pth roots can resume.

In computing terminology, the algorithm recursively calls itself on $S_0, T_1, T_2, \ldots$ to simplify them after deriving them as sums.

4.2 Special case: $q = 2$

When the factor q of f chosen is 2, the above algorithm simplifies and no multisums are needed to simplify A_0. We have $S_0 = A_0 + A_1$ and $S_1 = A_0 - A_1$, leading to $T_1 = (A_0 - A_1)^2$. Similar to the above, it can be shown that the expressions S_0 and T_1 have symmetry $2(p-1)/f$. Then A_0 can be written as $\frac{1}{2}(S_0 \pm \sqrt{T_1})$, where the sign can be determined by floating-point calculation.

4.3 Extension to multisums

The above algorithm is described as acting on sums of pth roots of unity for a single p. Because some of the simplified expressions returned by the algorithm are multisums, we need a way to simplify those too. In fact, the algorithm naturally extends to multisums. If A_0 is a multisum of p_1st, p_2nd, etc. roots of unity with a symmetry of $(p_1 - 1)/f$ with respect to p_1, and q is a prime factor of f, then expressions $S_0, T_1, \ldots, T_{q-1}$ can be derived. The expression S_0 is a multisum of p_1st, p_2nd, etc. roots, like A_0, with a symmetry of $q(p_1 - 1)/f$. The expressions $T_1, \ldots, T_{q-1}$ have qth roots added to the multisum and have symmetries of $q(p_1 - 1)/f$ with respect to p_1 and 1 with respect to q.

4.4 Base case: symmetry of $p - 1$

The above algorithm allows one to simplify a sum or multisum of pth roots of unity by recursively defining it in terms of expressions with increased symmetry with respect to p. Eventually we reach a point at which the symmetry becomes $p - 1$. At this point, the expression is of the form

$$A_0 = B_0 + B_1 x^1 + B_1 x^2 + \cdots + B_1 x^{p-1}$$

where B_0 and B_1 are sums or multisums of p_2nd, p_3rd, etc. roots of unity if A_0 is a multisum, or single integers if A_0 is not. As described in "Symmetric sums", we then have:

$$A_0 = B_0 - B_1$$

expressing A_0 without pth roots and reducing the sum's dimension by 1.

If A_0 was a single sum, not a multisum, it is now expressed as a single integer. This is the ultimate base case, because all integers in the final radical expression are derived in this way.

References

[1] Gauss, Carl F. *Disquisitiones Arithmeticae*. Yale University Press. pp. 359-360, 1965. ISBN 0-300-09473-6.

[2] Lynn, Ben. "Number Theory - The Heptadecagon." Stanford Applied Cryptography Group, `https://crypto.stanford.edu/pbc/notes/numbertheory/17gon.html`. Accessed August 17, 2019.

[3] Lynn, Ben. "Number Theory - Roots of Unity." Stanford Applied Cryptography Group, `https://crypto.stanford.edu/pbc/notes/numbertheory/rootsunity.html`. Accessed August 17, 2019.

[4] Lynn, Ben. "Miscellany - Roots of Unity." Stanford Applied Cryptography Group, `https://crypto.stanford.edu/pbc/notes/misc/rootsunity.html`. Accessed August 17, 2019.

[5] Weber, Andreas; Keckeisen, Michael. "Solving Cyclotomic Polynomials by Radical Expressions" (PDF). Accessed August 17, 2019.

[6] Weber, Andreas. "Computing radical expressions for roots of unity." *ACM SIGSAM Bulletin*, 30, 1996, pp. 11-20. 10.1145/240065.240070.

[7] Lau Jing Feng. "On solvable septics." *ScolarBank@NUS Repository*. January 7, 2005.